GOS 3.0.2
GSH User Manual

版本：0.3.4
作者：中国科学院计算技术研究所
时间：2008-4-29

GOS 3.0.2
GSH User Manual

版本：**0.3.4**

作者：中国科学院计算技术研究所

时间：2008-4-29

中国科学院计算技术研究所

目录

1. 概述 4
2. 安装gosclient 4
2.1 获取安装包 4
2.2 准备 5
2.3 安装步骤 5
3. 使用gsh 5
3.1 登陆 6
3.1.1 简单用法 6
3.1.2 高级用法 6
3.2 使用gsh 7
3.2.1 简单使用 7
3.2.2 执行网格命令 7
3.2.3 执行网格程序 19
3.2.4 执行本地命令或应用 19
3.2.5 执行.war文件 19
3.2.6 执行网格脚本 19
3.3 退出 20
3.4 语法和局限性 20
3.4.1 Java实现的gsh 20
3.4.2 C语言实现的gsh 21

1. 概述

本文档描述了 gosShell（简写 gsh）的安装和使用。gsh 为 GOS 提供了一个基于文本交互的用户使用界面，包含了应用运行环境和一系列相关的系统命令。目前 gsh 已经实现了两个版本，分别基于 java 语言和 C 语言。Java 实现的 gsh 可以跨平台，但是仅支持行命令，对脚本支持不足，而且缺少一些快捷提示等好用性支持；C 实现的 gsh 是对 GNU bash 的实现进行了一定的功能扩展，可以很好的保持用户原有的命令行使用习惯，但是不支持 windows 平台。

在安装完 GOS 软件之后，安装路径中可以看到一个 gosShell 目录，其中包含了 java 实现的 gsh，安装用户可以直接进入这个目录的 binary 目录下启动 gsh 的执行；在 GOS 的安装路径中还可以看到 GOS 客户端的安装包，gosclient.zip 用于 windows 平台，其中包含了 java 实现的 gsh 软件，gosclient.tar.gz 用户 linux 平台，其中包含了 java 实现的和 C 实现的 gsh 两个版本。用户可以根据操作系统的不同选择在不同的客户端安装包，在自己的客户端机或者网格节点的 home 目录下完成 gosclient 的安装过程。当然，这个安装过程也可以由某一个用户在一个网格节点的公共可访问的位置完成，供其他所有用户使用。

无论是 java 实现的 gsh 还是 C 语言实现的 gsh，从 gsh 的启动方式来讲，都分为如下几种情况：

- 如果安装在客户端机，则直接由用户启动${GSH_HOME} /binary/grun.sh 即可；
- 如果安装在网格节点的 home 目录，则用户需要通过 ssh 或者其他远程登录工具登陆到相应的网格节点后端机启动相应位置的 gsh 的执行；
- 如果安装在网格节点的公共可访问路径，则用户可以选择通过 ssh 或者其他远程登录工具登陆到相应的网格节点启动 gsh 的执行，也可以在客户端通过 GOS 提供的另外一个工具 VegaSSH 远程启动 gsh 的执行。

在本文档中暂不包含 VegaSSH 的使用手册。

下面将着重介绍 gosclient 的安装和 gsh 的使用过程。

2. 安装 gosclient

gosclient 可以安装在非网格节点，而通过 gosShell 管理和使用网格。

gosclient 也可以安装在公共的位置供多个本地用户使用（就像 bash 由 root 安装被所有 Linux 本地用户使用），也可以由某个用户安装在自己的 home 文件夹仅仅供自己使用。

2.1 获取安装包

在 GOS 的安装包中自带了 gosclient 的安装包。当 GOS 安装成功后，在 GOS 安装路径下的 all 或者 core 文件夹（当核心与系统分开安装时在 core 文件夹，一起安装时在 all 文件夹）中有 gosclient.tar.gz 和 gosclient.zip 文件，这就是与此版本相匹配的 gosclient 的安装包。其中 gosclient.tar.gz 是 Linux 发行版，gosclient.zip 是 Windows 发行版。两个版本的安装过程类似，这里以 linux 版为例讲述具体的安装步骤。

2.2 准备

gosclient 安装前需要预先做好如下准备：

- 预先安装 JDK1.5.0 及其以上版本（如果 gosclient 安装在公共位置供所有人访问的时候，jdk 也需要所有用户均可访问执行）并设置 JAVA_HOME 环境变量
- 预先安装 Apache ant 1.6.5 及其以上版本
- 预先安装 C 语言开发工具 gcc、yacc、autoconf、make（仅针对 linux 版安装）
- 安装 gosclient 的客户端机器必须能访问网格节点的 gos 服务端口
- 预留 40M 左右的磁盘空间

2.3 安装步骤

下面以 Linux 环境为例，给出 gosclient 的安装过程。

1. 获取安装包 gosclient.tar.gz，并存放在 gosclient 的安装目录。如果需要 gosclient 被多个 Linux 本地用户使用，请保证多个用户均能访问 gosclient 的安装目录。
2. 执行 tar zxf gosclient.tar.gz，解压 gosclient.tar.gz；解压后会在当前目录下生成名为 gosclient 的文件夹。
3. 进入 gosclient/installBase，对配置文件 gosconf.properties 中内容进行设置，可以设置的选项有：客户端链接到的网格节点 IP（gos_host）及 port（gos_port），客户端本地 log 日志文件存放路径（log_path）。
4. 执行 install.sh，安装 gos 客户端，例：
   # ./install.sh
5. 在安装过程中，若系统环境变量中未设置 JAVA_HOME 环境变量，则会提示用户手动输入 jdk 路径；若设置了 JAVA_HOME 环境变量则不需手动输入。如果希望 gosclient 被多个 Linux 本地用户使用，请保证从此 JDK 对所有用户可读。
6. 若 gosconf.properties 中未设置 log_path，也会提示用户手动指定一个 log 日志保存路径。如果希望 gosclient 被多个 Linux 本地用户使用，请保证从根目录/到 log_path 的全部路径上所有文件夹的权限为 777。
7. 安装完毕后，会在 gosclient 目录下生成出 certs，clientLib，conf，gosShell，logs 等文件夹，到此安装 gos 客户端成功。进入 gosShell 的 binary 目录下，执行 grun.sh 即可启动 gsh 的执行，具体在下一节介绍。

注意：确认在 conf 文件夹下的 namingserver.conf 文件中， localSiteUrl 属性指向正确的网格节点（该节点必须安装了 gos，并启动了 gos），IP 与 port 设置正确。也可手动修改指向其他网格节点。

注意：gosclient 中 clientLib 中的 jar 包必须和 GOS 服务端版本匹配，否则可能有一些无法预料的错误。如果是获取的与服务端版本一致的 gosclient 安装包，则这里不用关心此问题，否则请替换 clientLib 中的 jar 包。

3. 使用 gsh

本节使用用例，如不特殊说明，均以 Linux 系统中执行输入为例，windows 系统中执行

脚本和环境变量的表示等方面可能会有所不同。

3.1 登陆

当使用 VegaSSH 登陆 GOS 系统，登陆成功之后直接进入 gsh 的界面，无需本节介绍的登陆方式。如果用户已经通过类似ssh的方式登陆到服务器主机，此时可以选择直接启动gsh，启动 gsh 首先需要登陆。

3.1.1 简单用法

进入${GSH_HOME}/binary 目录，执行命令
> ./grun.sh
然后根据提示输入网格用户名和密码，即可实现登陆，进入 gsh 使用界面，这时启动的是 java 实现的 gsh，在这个 gsh 界面中可以直接敲入 ghelp 命令查看可执行的网格命令，也可以执行 GOS 命令或者系统本地的单行命令。
> ./grun.sh –c ./gsh
这个命令可以启动 C 实现的 gsh。

3.1.2 高级用法

usage: grun [xml_user xml_agora | -u user [-w password] -p proxy] [-a agoraname] [-c command | -ac command]

-a specify the login agora name
-c specify the command as grip instead of gsh
-ac specify the command as grip executed asynchronizely
-pf specify the login proxy file path name
-u specify the gos username
-p specify the gos user password

登陆过程可以带一些特定参数和选项，具体介绍如下：

[执行参数说明]

xml_user

带有用户身份的 `org.gos.core.grip.utils.UserAuthStruct` 对象的 xml 序列，指定启动 gsh 的网格用户身份。

xml_agora

带有社区信息的 `org.gos.core.agora.client.AgoraHandle` 对象的 xml 序列，指定登陆后所在社区。

xml_user 和 xml_agora 两个参数均用于通过编程方式获取到网格用户身份并启动 gsh 程序的情况，此时 login 程序将不再执行登陆行为，直接启动 gsh。

通常用户在命令行环境中直接执行 gsh 程序的时候无需使用这两个参数，主要提供给 VegaSSH 的服务端程序启动该程序时使用。

[执行选项说明]

-pf proxy_file

通过 proxy 方式登陆，用户需要在选项中制定 proxy 文件的路径和名字，grun 程序运行时将读取 proxy 文件登陆系统。

-u user_name

通过用户名密码方式登陆，通过 u 选项指定网格用户名，grun 程序运行时会提示输入密码，然后登陆系统。

如果同时指定了用户名和 proxy，如果两者不一致，也会导致登录系统不成功。

-p password

通过用户名密码方式登陆，当通过 u 选项指定网格用户名的同时，可以通过 p 选项指定用户密码登陆系统。

-a agoraname

通过 a 选项可以指定登录的社区名字，如果没有指定或者指定的社区登录出现错误（社区不存在或者没有非该社区用户），则直接登录用户的 home 社区。

-c command

通过 c 选项可以指定以网程的方式启动任意一个本地可执行程序，代替 gsh 的启动。

-ac command

通过 ac 选项可以指定以网程的方式启动任意一个本地可执行程序，代替 gsh 的启动，与-c 选项不同的是该选项异步启动命令的执行。

3.2 使用 gsh

3.2.1 简单使用

Java 实现的 gsh 和 C 实现的 gsh 中使用方式类似：

[gos] grun command_args

启动一个新的网程。如果 command_args 是一个带参数的命令，则执行相应网格命令或者应用，比如执行“grun java –Dgos.home=$GOS_HOME GagoraInfo”，gsh 会以网程的方式启动应用并执行查看当前网格中所有社区的基本信息。如果 command_args 是一个带路径的 war 文件，则部署相应的网格 web 应用，比如执行“grun test.war”，gsh 会找到 test.war，并以网程的方式部署到网格容器中。

[gos]command_args

执行系统本地命令或者应用，比如执行“pwd”，显示当前本地文件系统路径。

具体使用语法和支持的特定符号，在两个版本的 gsh 中有所不同，详见 3.4 功能和局限性。

3.2.2 执行网格命令

执行 ghelp 命令可以查看当前系统提供的可用命令名字，每个网格命令都带有-h 选项，给出命令具体的功能和使用格式，以及参数说明，目前 GOS 系统中提供的命令的具体功能和用法分别介绍如下：

社区相关(14 个)	**gadda, glsa, gdela, gchgainfo, gaproxy, gwa, gchga gaddg, glsg, gdelg, gchgginfo, gaddgu, glsgu, gdelgu**

用户相关(9 个)	**gaddu, glsu, gdelu, gchguinfo, glnu, gulnu, gupasswd, guproxy, gwu**
资源相关(7 个)	**gaddr, glsr, gdelr, gchgrinfo, glnr, gulnr, gexportr**
网程相关(2 个)	**gps, gkill**
系统相关(5 个)	**gnm, ghelp, galias, genvset, genvunset**

gadda　添加社区

在当前网格系统中添加一个新的社区，该命令没有任何参数，通过交互式的方式提示用户输入必要的信息，建立新的社区。任何一个合法的网格用户都可以新建社区。

使用格式：

```
usage: gadda [-h]
  -h     print help for the command
```

glsa　查看社区基本信息

查看满足指定条件的社区的基本信息。

使用格式：

```
usage: glsa [-h] [-i guid] [-n name][-o user][-l]
  -h      print help for the command
  -i      specify the guid of the agora
  -l      give a detailed information
  -n      specify the name of the agora
  -o      specify the owner name of the agora
```

gdela　删除社区

从当前网格系统中删除指定社区，命令会针对符合条件的社区逐一询问用户是否删除，用户确认之后才完成删除动作，-f 选项强制删除，不询问用户。如果删除某一个社区的时候出错，则给出错误信息并结束命令执行。

使用格式：

```
usage: gdela [-h] [-f] [agoraname] [-i agoraid] [-o ownername]
  -f      delete agora without prompt
  -h      print help for the command
  -i      specify the guid of the agora
  -o      specify the owner name of the agora
```

gchgainfo　修改社区元信息

修改社区元信息，具体使用方式和 GchgResAttr 命令相同，针对用户只有 ACL, Description, 和 Attributes 可以被修改，如果命令参数中没有指定社区名或者社区 id，则修改当前社区的元信息。

该命令用于修改 GOS 社区的元信息，agoraname 参数和-i 选项用于指定一个目标社区，如果都省略则目标社区为当前社区。如果有多个社区对应，命令会提示用户选择其中一个进行修改动作。-n 选项指定修改的元信息名字和修改值，如果没有-n 选项，则命令会将目标社区的所有元信息列出，同时给出一个简单的命令控制台和简单的控制台使用方法，如下：

only ACL, Group, Description and Attributes can be modified
'help' to get command help
'name=value' format to change some info
'remove name' format to remove the attr
'submit' to make modification effect
'list' to get current info
'squit' to exit with saving
'quit' to exit without saving

这里说明了对于社区，只有 ACL, Group, Description 和 Attributes 是可以修改的，其他 gnode 基本信息不支持用户任意修改，help 将给出这个小控制台的简单使用格式，name=value 设定了某个信息值，remove name 删除特定名字的属性信息，submit 将当前的修改提交给系统正式修改，list 列出当前修改后的值，squit 将当前的修改提交给系统同时退出，quit 直接退出当前命令返回到 gsh 中。

使用格式：

usage: gchgainfo [agoraname] [-i agoraid] [-n AttrName=newValue] [-h]
-h　print help for the command
-i　specify the guid of the agora
-n　modify info with the format: AttrName=newValue

gaproxy　更新社区 proxy

更新社区的 proxy，如果省略社区名字则更新当前社区的 proxy。新的 proxy 中指定的 DN 需要和社区中原来的 DN 保持一致。

使用格式：

usage: gaproxy [-h]　[aroganame] [-i agoraid] proxyfile
-h　print help for the command
-i　specify the guid of the agora

gwa　查看当前社区

查看 gsh 环境中用户当前所在的社区的基本信息。

使用格式：

usage: gwa [-h]
-h print help for the command

gchga 改变当前社区

该命令目前是 gsh 的内部命令，更改用户当前所在社区位置。如果不带任何参数，则切换到用户的默认社区，否则切换到用户指定的社区。如果用户不是目标社区的用户或者目标社区不存在，则返回出错信息，保持原社区上下文信息，否则提示社区已经切换成功的信息。
使用格式：

usage: gchga [-h] [aroganame]
-h print help for the command

gaddg 在当前社区中增加用户组

在当前社区中增加一个用户组。如果不带 groupname 参数，命令会提示用户输入，并根据用户输入信息建立用户组。
使用格式：

usage: gaddg [-h] [groupname]
-h print help for the command

glsg 查看当前社区的组信息

查看当前社区中所有用户组的基本信息。通过-n 选项可以查看指定名字的用户组信息，-u 选项可以查看指定用户所在的用户组信息。
使用格式：

usage: glsg [-h] [-n groupname] [-u username] [-l]
-h print help for the command
-l list a detailed information
-n list the given name 's group info
-u list the groups where the given user is

gdelg 当前社区中删除用户组

从当前社区中删除指定名字的用户组。
使用格式：

usage: gdelg [-h] groupname
-h print help for the command

gchgginfo 修改指定用户组的信息

修改执行用户组的信息，目前仅支持修改用户组的描述信息。

使用格式：

usage: gchgginfo [-h] groupname [-d newdescription]

-d specify new scription of the group

-h print help for the command

gaddgu 把用户加到某个用户组中

把当前社区中存在的特定用户增加到社区中的特定用户组中。

使用格式：

usage: gaddgu [-h] username groupname

-h print help for the command

glsgu 查看用户组中的用户

查看指定用户组中存在哪些用户。这个命令必须在用户组所在的社区中完成。

使用格式：

usage: glsgu [-h] [groupname] [-l]

-h print help for the command

-l give a detailed information

gdelgu 从用户组中删除指定用户

从指定用户组中删除指定用户。这个命令必须在用户组所在的社区中完成。

使用格式：

usage: gdelgu [-h] username groupname

-h print help for the command

gaddu 注册用户

该命令用于在当前社区中增加一个已经拿到证书 proxy，但是没有注册到网格中的用户。该命令没有任何参数，执行的时候按照提示输入相关信息即可。

使用格式：

usage: gaddu [-h]

-h print help for the command

glsu　查看用户信息

查看当前社区中用户的基本信息，节点管理员可以通过-r GRID 选项查看全网格系统中符合条件的用户基本信息。-l 选项流出详细信息。

使用格式：

usage: glsu [-h][-n username] [-r GRID|AGORA] [-l]

-h　print help for the command

-l　give a detailed information

-n　specify the user name

-r　specify the search range: GRID or AGORA(default)

gdelu　删除用户

从当前社区中删除指定的用户，命令执行后会提示用户是否确认删除，-f 选项强制删除，不给提示。如果当前社区是加目标用户的初始社区，则该命令会把这个用户从网格系统中删除，如果当前社区不是加目标用户的初始社区，则该命令仅把用户从当前社区中删除，用户仍然存在在网格系统的其他社区中。

使用格式：

usage: gdelu [-h] username [-f]

-f　delete user without prompt

-h　print help for the command

gchguinfo　修改用户元信息

修改用户元信息，具体使用方式和 gchgainfo 命令类似，如果命令参数中没有指定用户名，则修改当前用户的元信息。

使用格式：

usage: gchguinfo [-h] [username] [-n AttrName=newValue]

-h　print help for the command

-n　modify info with the format: AttrName=newValue

glnu　增加网格用户到当前社区

该命令用于要把存在于其他社区的指定网格用户加入到当前社区中。

使用格式：

usage: glnu [-h] username

-h　print help for the command

gulnu 从当前社区中删除连接用户

该命令用于把当前社区中的连接用户删除。如果是原始用户信息，需要使用 gdelu 删除。

使用格式：

usage: gulnu [-h] username [-f]
-f delete user without prompt
-h print help for the command

gupasswd 修改用户密码

修改用户的密码，如果省略用户名参数，则修改当前用户的密码，命令执行后会提示输入新密码并确认输入，两次输入一致后更新用户密码。

使用格式：

usage: gupasswd [-h] [username]
-h print help for the command

guproxy 更新用户 proxy

更新用户的 proxy，如果省略用户名参数，则更新当前用户的 proxy，更新的 proxy 中的用户 DN 必须和目标用户的 DN 保持一致。

使用格式：

usage: guproxy [-h] [username] proxyfile
-h print help for the command

gwu 查看当前用户

查看当前 gsh 环境中的网格用户身份。如果当前用户元信息被修改，会在下次登陆中体现出来。

使用格式：

usage: gwu [-h]
-h print help for the command

gaddr 当前社区中增加资源

在当前社区中增加一个 GOS 资源，该命令没有任何参数，命令交互式执行，用户通过提示完成相应信息的填写即可增加一个自定义的资源。

使用格式：

usage: gaddr [-h]
-h print help for the command

Glsr　查看资源信息

查看资源基本信息，默认查看当前社区的有所资源基本信息，节点管理员可以通过-r GRID 选项查看整个网格系统所有资源的信息，其他各个选项给出资源筛选的条件。

使用格式：

```
usage: glsr [-h] [ [-i guid][-n name] [-t type] [-o user] [-o1 introducer]
                | [-c conditions] ] [-l] [-r GRID|AGORA]
  -c      specify customCondition of the resources
  -h      print help for the command
  -i      specify the guid of the resource
  -l      give a detailed information
  -n      specify the name of the resource
  -o      specify the owner name of the reousrce
  -o1     specify the introducer of the resource
  -r      specify the search range: GRID or AGORA(default)
  -t      specify the controller name of the resource
```

选项-c conditions 的格式如下：

```
logicalOrExpression
    :   logicalAndExpression ("or" logicalAndExpression)*
    ;
logicalAndExpression
    :   unaryExpression ("and" unaryExpression)*
    ;
unaryExpression
    :   "not" unaryExpression
    |   primaryExpression
    ;
primaryExpression
    :   relationExpression
    |   "(" logicalOrExpression ")"
    ;
relationExpression
    :   nameString op valueString
    ;
op
    :   "<" | ">" |   "<=" | ">=" | "=" | "!=" | "like"
    ;
```

gdelr　删除资源

从当前社区中删除目标资源。命令会针对给定的一个或多个目标资源逐一由用户确定是否删除，根据用户的反馈实施具体的动作，如果指定了-f 选项，则命令直接将符合条件的所

有目标资源全部删除。

使用格式：

usage: gdelr [-h] [-f][-i guid][-n name] [-t type] [-o user] [-o1 introducer]

-f delete resource without prompt

-h print help for the command

-i specify the guid of the resource

-n specify the name of the resource

-o specify the owner name of the reousrce

-o1 specify the introducer of the resource

-t specify the type of the resource

gchgrinf 修改资源元信息

修改资源元信息，具体使用方式和 gchgainfo 命令类似。

使用格式：

usage: gchgrinfo [-h] [resname] [-i guid] [-n AttrName=newValue]

-h print help for the command

-i specify the guid of the resource

-n modify info with the format: AttrName=newValue

glnr 导入资源到当前社区

将其他社区的资源导入当前社区中，相当于 linux 系统中建立一个符号文件。目前由于普通用户无法通过名字查询其他社区的资源，推荐直接通过-i 选项指定目标资源。

使用格式：

usage: glnr [-h] [resname] [-i resguid] [-f agoraname]

-f the orignal agora name of the resource

-h print help for the command

-i specify the guid of the resource

gulnr 从当前社区中删除链接资源

该命令用于把当前社区中的连接资源删除。如果是原始资源信息，需要使用 gdelr 删除。

使用格式：

usage: gulnr [-h] [-f][-i guid][-n name] [-t type] [-o user] [-o1 introducer]

-f delete resource without prompt

-h print help for the command

-i specify the guid of the resource

-n specify the name of the resource

-o specify the owner name of the reousrce

-o1 specify the introducer of the resource
-t specify the controller name of the resource

gexportr 设置其它社区对该资源的导入权限

设置其他社区对当前社区某个指定的资源的导入权限。resname 参数和-i 选项指定了目标资源，-u 选项取消指定社区对目标资源的导入权限，-a 增加某社区对目标资源的导入权限，-s 重新设置该权限。这三个选项后面带参数的格式如下：

```
agoralist:
    agoraitem (, agoraitem)*
    ;
agoraitem:
    i:guid | name | * | i:guid:s | name:s| *:s
    ;
```

其中 i 表示后面是一个社区的 guid，s 表明以 self 模式允许该社区导入，导入后资源仍然由原资源宿主管理，如果不加 s 代表以 delegate 模式允许该社区导入，导入后资源允许被社区管理员管理。

使用格式：

```
usage: gexportr [-h] [resname] [-i resid] [-u agoralist] [-s agoralist]
                [-a agoralist] [-l]
```

-a add these exports
-h print help for the command
-i specify the guid of the resource
-l check current export set
-s set these exports
-u cancel these exports

gps 查看当前运行的网程

查看当前节点当前用户启动的所有网程基本信息，可以通过-s 指定查看特定节点的网程信息，如果不是默认端口，需要通过 IP:port 格式指明端口号。节点管理员可以通过-u 选项查看指定用户的网程信息。

使用格式：

```
usage: gps [-h] [-l] [-u username] [-t java-apps|web-apps] [-s siteIP]
```

-t grip type, (java-apps|web-apps)
-h print help for the command
-l print a detailed information
-s give site ip where grip hosted
-u print grip of the given username, default is current user, * represents all users

gkill 杀网程

强制终止指定网程的执行。不支持普通用户终止非自己启动的网程。

使用格式：

usage: gkill [-h] gripID

-h print help for the command

ghelp 查看可用命令的使用说明

该命令用于列出所有 gsh 中自带的命令。-t 可以列出所有的命令组，带一个参数可以输出指定命令组中的所有命令，默认按照命令组输出所有可用的 GOS 命令。

使用格式：

usage: ghelp [-h][-t | typename]

-h print help for the command

-t list command types

galias 查看设置命令别名

（仅在 Java 实现的 gsh 有效）该命令目前是 gsh 的内部命令，用于维护 gsh 内部别名设置。gsh 别名类似 GNU bash 中提供的别名机制，用户定义的别名值将替换相应命令的第一个 token。gsh 启动的时候会把${GSH_HOME}/conf/command.alias 文件中定义的别名导入系统，在 gsh 执行过程中可以通过该命令对别名定义进行维护，-f 选项从文件中导入别名定义增加到当前 gsh，直接带 name=value 参数将该别名定义增加到当前 gsh，-r 选项如果和-f 选项同时使用，将从当前 gsh 中删除指定文件中的别名定义，如果和 name=value 参数一起使用，则把参数整个字符串认为一个别名，从 gsh 环境中删除。

使用格式：

usage: galias [-h] [-r] [name=value | -f aliasfile]

-f specify alias defintion file

-h print help for the command

-r remove the specified alias definition

genvset 设置网格环境变量

该命令用来管理 gsh 中的网格环境变量。在网格上下文中，允许用户自定义网格环境变量，并可以设置到新启动的网程中，或者随着网程继承。如果参数不是 name=value 模式，则为查看指定环境变量名字的值，如果没有任何参数，则显示所有已经设置的网格环境变量的值。

使用格式：

usage: genvset [-h] [name=value | name]

-h print help for the command

genvunset 取消网格环境变量定义

该命令用来取消网格环境变量的某项定义。

使用格式：

usage: genvunset [-h] name

-h　　print help for the command

gnm　管理网络拓扑

该命令用来管理网络拓扑以及节点的基本信息，一般仅供节点管理员或者网格管理员使用。这个命令是根据具体的 command 来执行相应的操作。主要的 command 包括三类：

网格的拓扑管理：createNetwork，join，leave，list，listNb

本节点的 naming 管理：isStarted，start，stop，restart

本节点的配置信息：localInfo，loadConfig

使用格式：

usage: gnm command parameters [-n namingUrl] [-l] [-h]

-h　　print help for the command

-l　　give a detailed information

-n　　specify the naming URL

所支持的 command 具体语义如下：

- createNetwork networkName　用来建立网格。首先由一个节点执行 createNetwork。运行时必须指定 networkName。比如 gnm createNetwork cngrid 建立一个名叫 cngrid 的网格。
- getEvents siteId startSn
- help　给出帮助信息
- isStarted 本节点的 naming 的后台处理线程是否在运行
- join bootstrapURL　用来将节点互联到已经建好的网格，执行时指明一个已经在此网格中的节点的 naming 的 URL 作为 bootstrapURL。比如 gnm join http://10.61.0.45:38080/axis/services/naming
- leave　将本节点离开网格
- list　列出当前网格的全部节点
- listNb　列出当前节点的全部邻居
- loadConfig　查看本节点的管理员、默认用户等配置信息
- localInfo　查看本节点的基本信息
- locate guid 查看特定 guid 的 Gnode 的信息。
- restart 重启 naming，此时$gos.home/conf/namingManager.conf 中修改过的配置将会生效
- search searchCondition values 根据条件搜索 Gnode。目前只支持 searchCondition 和 values 都为”null”时搜索所有的 Gnode。
- start　启动 naming
- stop　停止 naming
- sync remoteSiteId 将 remoteSiteId 所指定的节点的 Gnode 同步到本地。

- unregister guid 注销 guid 指定的 Gnode。强烈不推荐使用，否则可能引起数据不一致。
- viewSite siteID 查看 siteID 指定的节点的信息。

3.2.3 执行网格程序

两种版本的 gsh 中均已经正确设置了 GOS_HOME，CLASSPATH，LD_LIBRARY_PATH 环境变量，以方便网格应用执行，因此在 gsh 环境中，输入

[gos]grun command args

即可启动相应的网格应用执行。比如

[gos]grun ./myapp hellomsg

如果是 java 程序，需要首先设置实现类包含在 CLASSPATH 环境变量中，启动应用的时候设置 java 虚拟机的 gos.home 属性（可以通过环境变量设置），比如

[gos]export CLASSPATH=$CLASSPATH:./myapp.jar

[gos]grun java –Dgos.home=$GOS_HOME MyClass

在 gsh 环境之外，比如 GNU bash 中，也可以通过 grun.sh 启动执行网格应用。可以通过-u 和-p 参数（或者-pf 参数指定 proxy 文件）分别给定用户身份，以及-c 或者-ac 选项带执行的命令参数即可，详见 3.1.2 登陆部分的高级用法。这样的命令可以嵌入到任何本地执行脚本（比如 sh 脚本）中被执行。

3.2.4 执行本地命令或应用

别名替换之后，没有以 grun 开头的命令认为是本地执行的命令或者应用。比如

[gos]vi testfile.in

调用了本地系统命令执行编辑文件的操作。

gsh 中本地支持语法格式详见 3.4 的局限性。

3.2.5 执行.war 文件

两个版本的 gsh 环境中，

[gos]grun warfilename

即可完成 web 应用类以网程方式的启动和部署，不支持任何参数。比如

[gos]grun ./mywebapp.war

3.2.6 执行网格脚本

Java 实现的 gsh 中不支持脚本功能，仅支持单行简单命令的执行。

C 实现的 gsh 中兼容 GNU bash 原有的脚本定义，在这样的脚本中如果插入了网格命令或者应用的执行，我们称这样的脚本为网格脚本。一个网格脚本文件，需要脚本的第一行的内容为“#![gsh 可执行文件名]”这样的 magic number，然后使用“grun [脚本文件名]”的方式执行脚本。这样可以保证网格脚本中启动的网格应用能够正确的执行。比如

```
[gos]cat myScript.gsh
#!./gsh
grun java –Dgos.home=$GOS_HOME GuserInfo | grep wxn@software.ict.ac.cn
……
[gos]grun ./myScipt.gsh
```

3.3 退出

两个版本的 gsh 均通过 gexit 命令退出 gsh 程序。在 C 实现的 gsh 中，还可以通过输入 exit 或按 Ctrl-D 快捷键退出 gsh 程序。

3.4 语法和局限性

3.4.1 Java 实现的 gsh

目前 Java 实现的 gsh 命令执行仅支持单行单个简单命令或者应用，不支持任何本地 sh 脚本的网程方式执行，也不支持网程程序的后台运行。shell 中常见的输出输出重定向，管道，以及执行控制脚本功能将在今后的工作的逐步考虑支持。Gsh 针对 linux 系统平台或者 windows 系统平台分别实现了一套简单的词法分析，用于解析单行命令，具体支持如下：

Linux 系统

- 支持管道和简单重定向（| > < << >>），并保持原语义
- 支持&，保持原语义，但只允许出现在行最后一个字符
- 不支持分号（；）和大中小括号（ ( ) [ ] { } ）
- 双引号（“ ”）内的特殊符号被处理，如果不出现匹配的引号提示下一行继续直至匹配为止
- 单引号（‘ ’）内的特殊符号不被特殊处理，如果不出现匹配的引号提示下一行继续直至匹配为止
- 回引号（` `）内的字符串会被执行，如果不出现匹配的引号提示下一行继续直至匹配为止
- 反斜杠（ \ ）保持在 GNU bash 中的语义，即引号内语义转义，行末表示本行未结束
- 支持环境变量 （$）
- 对于符号~，当它本身是一个词或者以~/开始的词，~代表 home 目录字符串，其他认为非特殊字符
- 实现了 cd 和 export 两个内部命令，支持其他不更改当前 shell 状态的内部命令

Windows 系统

- 支持简单重定向（> < << >>），并保持原语义
- 不支持管道（ | ）和后台运行符（&）
- 不支持分号（；）和大中小括号（ ( ) [ ] { } ）
- 双引号（“ ”）内的特殊符号被处理，如果不出现匹配的引号，则引号内容到行末结束

- 单引号（‘’），回引号（``）和反斜杠（ \ ）不作为特殊字符处理
- 支持环境变量 （%%）
- 实现了 cd 内部命令，支持其他不更改当前 shell 状态的内部命令

对于 grun 方式执行的网格命令，不支持其与管道（ | ）和重定向（ > < << >>），以及后台运行（&）的组合使用。其他情况根据运行在不同的操作系统，遵循不同系统上的词法分析规则。

3.4.2 C 语言实现的 gsh

C 语言实现的 gsh 是基于对 GNU bash 的功能扩展，在其中增加了网格上下文维护和网程启动的功能，这两个功能和 bash 原有的功能和脚本支持都可以很好的结合在一起使用，目前经过测试发现存在如下几条限制：

- grun 执行起来的网格命令或者应用不支持 Ctrl-Z 挂起

如果这种方式启动起来的应用接收了 Ctrl-Z，会调度到后台执行，但是该执行不会被作业管理所控制，也就是说无法调度到前台执行，或者接收作业控制的相关命令和信号。

- gchga，genvset，和 genvunset 不可以与“|”、“&”、“()”、“``”同时使用

这样使用会导致进程挂起，不能正常执行的现象。

- 非交互执行方式（比如 gsh 脚本）默认不支持使用 glsr、gwa 等别名

在非交互 shell 下，gsh 不开启别名特性，因此在 gsh 脚本中 glsr、gwa 之类的别名无法使用。但是可以在 gsh 脚本中增加如下几行说明：

```
shopt -s expand_aliases
. $GSH_HOME/conf/cmd_alias_c.init
. $GSH_HOME/conf/command.alias
```

这样，随后的脚本中就可以继续使用 glsr 这样的脚本定义了。

另外，比如执行 glsr|more 命令，如果刚显示了一两页，就按“q”键退出，此时光标在闪，但没有 bash 提示符出现，过十几秒之后才出现 gsh 提示符。这是正常现象，不是死循环。因为 glsr 执行较慢，大约 30 秒才能返回所有输出。在 glsr 返回所有输出之前按“q”键退出 more，事实上 glsr 还在执行，必须等它执行完，gsh 提示符才会出现。这个现象在 GNU bash 中也是存在的，对普通的执行慢的程序，比如 find，在原版 bash 下测试：find / -name dev 2> /dev/null |more，现象和 glsr|more 一样。